IN THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

Claims 1-10 (cancelled)

11. (currently cancelled) A system comprising:

a portal server to retrieve classfiles from an Internet site on behalf of a data processing device, the data processing device comprising a processor for processing program code and an interpreter module for interpreting classfiles, the portal server communicatively coupled to the data processing device over a network;

wherein the portal server comprises a content conversion module to analyze and convert the classfiles prior to transmission to the data processing device over the network, the conversion comprising rearranging elements of two or more classfiles to form a unified programming object, the elements including a plurality of constant pool entries and one or more methods and/or fields; and

wherein after the portal server generates the unified programming object, it transmits the unified programming object to the data processing device over the network.

12. (Previously Presented) The system as in claim 11 wherein, to rearrange the elements of the two or more classfiles, the content conversion module combines redundant constant pool entries from said two or more classfiles to form a global constant pool entry in a shared constant pool within said unified programming object.

13. (Previously Presented) The system as in claim 12 wherein combining further comprises:

rewriting said global constant pool entries to point to elements contained within said unified programming object, said elements corresponding to elements contained in said one or more class files and previously identified by said one or more redundant constant pool entries.

14. (Previously Presented) The system as in claim 13 wherein one of said global constant pool entries is a methodref entry and said element identified by constant pool entry is a method copied to said unified programming object from said one or more class files.

15. (Previously Presented) The system as in claim 14 wherein, to rearrange the elements of the two or more classfiles, the content conversion module converts numeric references to local entries within a bytecode in said method to pointers to global constant pool entries.

16. (Previously Presented) The system as in claim 15 wherein, to rearrange the elements of the two or more classfiles, the content conversion module converts an exception table associated with said method to references to jop objects instead of numeric references to addresses of bytecodes.

17. (Previously Presented) The system as in claim 13 wherein one of said global constant pool entries is a fieldref entry and said element identified by constant pool entry is a field copied to said unified programming object from said one or more class files.

18. (Previously Presented) The system as in claim 11 wherein, to rearrange the elements of the two or more classfiles, the content conversion module validates said two or more class files before mapping said elements to form said unified programming object.

19. (Previously Presented) The system as in claim 18 wherein, to rearrange the elements of the two or more classfiles, the content conversion module converts the classfiles into a graph of jop objects to track where jump operations pointed before modification of said classfiles; adjusts constant pool references from local to global numbers based on said graph; and combines the classfiles into the unified programming object.

20. (New) The system as in claim 11 wherein the portal server automatically retrieves classfiles from the Internet site on behalf of the data processing device in response to requests for the classfiles from the data processing device.

21. (New) The system as in claim 11 wherein the portal server manually retrieves classfiles from the Internet site on behalf of the data processing device in response to requests by the organization managing the portal server.

22. (New) A method for converting a plurality of classfiles into a unified classfile bundle comprising:

generating a global method entry within the unified classfile bundle for each method within the plurality of classfiles, each global method entry containing data related to the method from which it was generated;

generating a global field entry within the unified classfile bundle for each field within the plurality of classfiles, each global field entry containing data related to the field from which it was generated;

converting local constant pool entries from local constant pools of each of the plurality of classfiles which reference methods and fields of each classfile into global constant pool entries within the unified classfile bundle, wherein converting comprises removing redundant local constant pool entries and replacing the references to methods and fields within each of the local constant pool entries with offsets pointing to the global method entries and field method entries, respectively, within the unified classfile bundle.

23. (New) The method as in claim 20 wherein converting further comprises converting variable-length local constant pool entries from each of the classfiles with fixed-length constant pool entries within the unified classfile bundle.

24. (New) The method as in claim 20 wherein converting further comprises:

for each local constant pool entry, determining whether a corresponding global constant pool entry already exists, and

if a corresponding global constant pool entry already exists, then using the existing global constant pool entry in lieu of the local constant pool entry; and

if the global constant pool entry does not exist, then generating a new global constant pool entry from the local constant pool entry.

25. (New) The method as in claim 20 wherein each global method entry comprises a Method Info object, and wherein generating the Method Info object comprises directly copying data of a first type from each method of each classfile and modifying data of a second type prior to copying the data from each method of the classfile.

26. (New) The method as in claim 23 wherein the data of the first type includes a reference to the method name via the constant pool; a reference to a type signature via the constant pool; access or attribute flags; max stacks data; max locals data; an argument count, and/or a return count.

27. (New) The method as in claim 24 wherein the data of the second type includes Method bytecodes, exception tables, and/or virtual table slot numbers associated with the method.

28. (New) The method as in claim 20 wherein each global field entry comprises a Field Info object, and wherein generating the Field Info object comprises directly copying data of a first type from each field of each classfile and modifying data of a second type prior to copying the data from each field of the classfile.

29. (New) The method as in claim 26 wherein the data of the first type includes a reference to the field name via constant pool; a reference to the field type signature via the constant pool; and/or any access/attribute flags.

30. (New) The method as in claim 20 further comprising:

generating a Class Info object within the unified classfile bundle for each header of each classfile, each Class Info object containing header data from each classfile header.

31. (New) The method as in claim 28 further comprising:

generating a relative pointer for each global method entry and each global field entry, the relative pointer pointing to a Class Info object that the global method entry or global field entry is associated with.

32. (New) The method as in claim 20 further comprising:

generating a relative pointer for each global method entry and each global field entry, the relative pointer pointing to a global constant pool entry that the global method entry or global field entry is associated with.